

## APPLICATION OF MACHINE LEARNING MODEL TO MICROCONTROLLERS - AUTOMATION OF IOT EDGE DEVICES

**Author's Name:** <sup>1</sup>MSc. Vo Hung Cuong, <sup>2</sup>MSc. Dinh Thi My Hanh, <sup>3</sup>Mr. Tran Cong Danh

**Affiliation:** <sup>1</sup>Lecturer Of Vietnam Korea University of Information and Communication Technology - The University of Danang

<sup>2</sup>The University of Danang, PhD student of Hanoi University of Science and Technology, Vietnam

<sup>3</sup>Student, Vietnam Korea University of Information and Communication Technology - The University of Danang

**E-Mail:** [vhcuong@vku.udn.vn](mailto:vhcuong@vku.udn.vn)

**DOI No. – 08.2020-25662434**

### Abstract

The Internet of Things has advanced at a breakneck pace in recent years. As a result, cloud servers are storing billions of records, causing delays for some IoT systems, which must transport data from many devices to the server and execute machine learning computations. As a result of the rapid growth of microcontrollers, a new idea known as edge computing was formed. Tensorflow lite is a big library that allows microcontrollers to employ machine learning models. In this post, we'll develop a system that uses a machine learning model placed on the ESP32 microcontroller to autonomously control lights and fans based on sensors in the surroundings. The Arduino Integrated Development Environment is utilized with TensorFlow Lite for Microcontrollers. With a varied number of neurons, neural networks with two hidden layers are employed.

**Keywords:** Internet of Things, Machine Learning, Microcontroller

### INTRODUCTION

Currently with the development of technology, most architectures of a popular IoT system consist of 3 key components:

- The Upper Layer (Cloud Layer) is a big data platform that analyzes complex and time-consuming tasks like Big Data, Machine Learning, etc.
- The middle layer is edge computing: The edge computing layer can be seen right next to or near IoT devices to connect and process the local data of billions of IoT devices. The term "edge computing" is used to describe computing centers located in the middle of the cloud but near devices, called marginals. It also depicts the anticipated boundary between internet and local network settings.. Some documents or articles that use the term "Boundary Computing" or "Edge Computing" may sound more unsymportable or confusing than the term "edge computing."
- The last layer is IoT devices: sensors, instrumentation, controllers, etc.

Previous projects, machine learning models were applied and developed to store and process billions of data on large server systems. By reading data from sensors in the border area to the server for storage, there will be some limited problems such as bottlenecks or bandwidth consumption. It will be very difficult and dangerous if the system uses a vertical machine to control the automated systems. That's why the computer was born to solve the problem above. By processing tasks and data right at a small center on the edge of the devices, all operations of the devices at the edge are through a local network with wireless communication protocols such

as radio waves. A computer system that is too huge and complicated does not require edge computing. A phone, a computer, or even a little microcontator might be used. Companies and enterprises will save a lot of money as a result of this.

In this topic, We used tensorflow lite to install and develop on the ESP32 micro-control module, which is programmed on IDE VSCode and platformIO environment. The sensor buttons are built with arduino promini, all communicated with the ESP32 gateway by the RF24 radio circuit. The turn on and off lights and fans is done through two modes monitored by the mobile app, the default gateway is done automatically using the machine learning model to predict, and the manual on and off mode is performed by the mobile app. The following is how the article is structured: Part 2 walks you through the process of designing a system and creating datasets. Part 3 builds a detailed program and part 4 concludes and evaluates.

### LITERATURE REVIEW

Currently, research and development to bring edge computing to life has been and is receiving much more attention. Typical of the AI revolution in the auto industry [1]. Many major car manufacturers are working to create their own self-driving cars and driving features. Regarding autonomous vehicles, the control system must be independent of the server, because it requires accuracy and fast data processing speed, which is called edge computing. The authors in [2] have applied artificial intelligence to develop a smart mirror, they have used facial recognition to help it work, in addition it can also manage the work schedule, control smart home, answer difficult questions. In the article [3] the authors relied on edge computing to develop a garbage classification model on an embedded board, this model is used car and camera to use to identify types of garbage.

### DESIGN OF THE SYSTEM

#### a. Model system node sensor

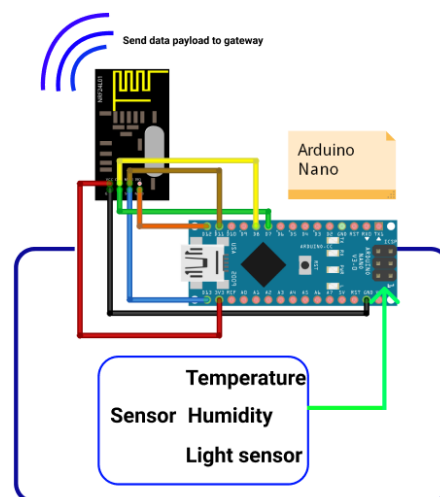


Figure 1 Model system node sensor

#### Principles of operation of the button:

Read sensor data (humidity, temperature, ambient light) => Arduino processing packing => sent via RF communication to the gateway address.

**b. Count the number of people**

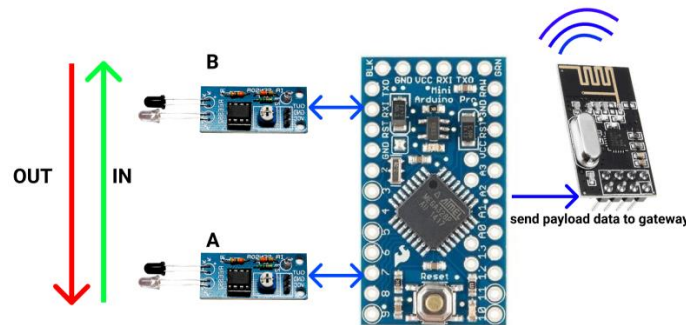


Figure 2 Model system count the number of people

**Operating principles:**

The system for checking people in and out of the room is located at the door of the room in order of infrared distance sensor A - B. When someone enters room sensor A is read first then to sensor B. Conversely when someone comes out, the B sensor will be read first then to sensor A.

**c. Node switch device**

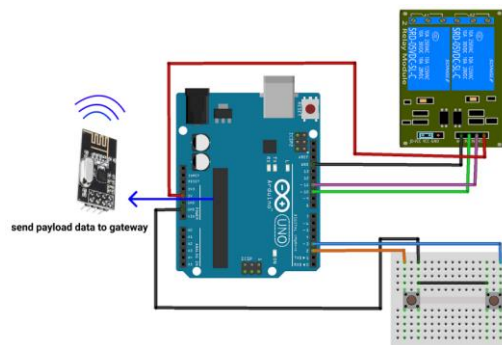


Figure 3 Model node switch device

To get the dataset for training models, this button will combine with sensors to produce a training dataset. Accordingly, when someone enters at the button to count the number of people entering and leaving the room, if it is dark, turn on the light at the push button or high temperature can turn on the fan. On the other hand, if there are no people in the room, the lights and fans will be turned off. Any on-off action at this button is combined with other button sensor data to add to the dataset table.

**d. Get dataset**

Use the ArduSpreadsheet tool to read from serial arduino IDE

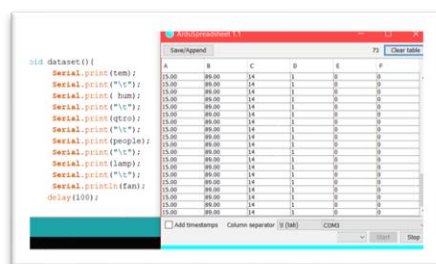


Figure 4 Get data from tool arduino

	A	B	C	D	E	F	G
8329	26	90	51	1	1	0	
8330	26	90	51	1	1	0	
8331	26	90	51	1	1	0	
8332	26	90	51	1	1	0	
8333	26	90	51	1	1	0	
8334	26	90	51	1	1	0	
8335	26	90	51	1	1	0	
8336	26	90	51	1	1	0	
8337	26	90	51	1	1	0	
8338	26	90	51	1	1	0	
8339	26	90	51	1	1	0	
8340							

Figure 5 Table dataset

**The result:** The dataset consists of 8340 rows and 6 columns

**e. Model project system**

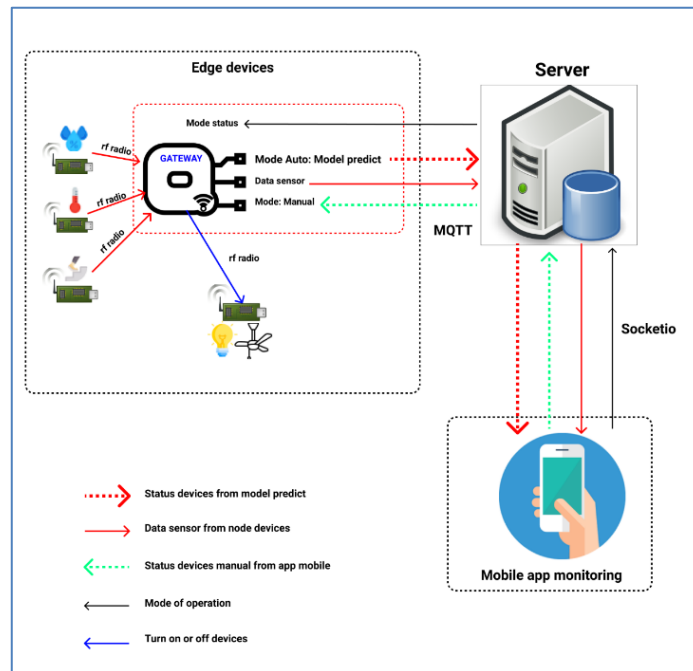


Figure 6 Model project system

The system consists of three components: The main part is the marginal devices consisting of a gateway that predicts on-off actions based on sensor data collected by nodes in the border area. The second part is the server with the function of storing environmental sensor data. The last part is the mobile app with the function of monitoring the parameters from the sensor and manually turning on and off the lights and fans. The cooling model is done on google colab.

**f. Building a machine learning model**

**Data processing**

The data before training will be separated into three parts:

- Data train: used for training during training
- Data test: used for testing after training the model
- Data validation: used for testing during training

```
X_train_val, X_test, y_train_val, y_test
= train_test_split(X, y, test_size = 0.3 )
X_train, X_val, y_train, y_val
= train_test_split(X_train_val, y_train_val, test_size = 0.2)
```

With X are input data fields that include:

Temperature	Humidity	Light level	Detect people
27.8	88.9	14	1
.....	.....	....	.....

Y includes lable output including:

- Lights and fans: *on (1), off (0)*

### g. Neurons

The DNN is made up of several processing layers that may extract hierarchical properties from the input data [4]. DNN's operation is modeled after that of the human brain. Each layer in the DNN consists of multiple processing units called neurons. A neuron performs the weighted sum of the inputs  $[x_1, x_2, x_n]$  and feeds the resulting sum to an activation function that generates the desired output. Each neuron consists of a set of weights  $[w_1, w_2, w_n]$  and a bias ( $b$ ) that is optimized during the training process.

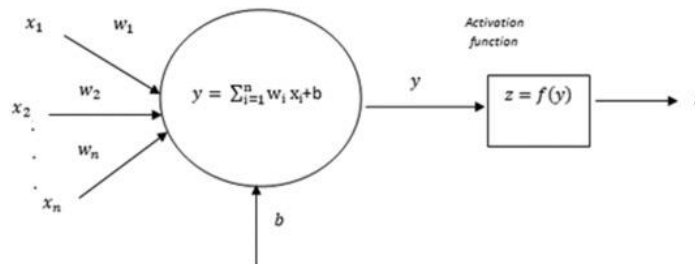


Figure 7 Processing unit

**In a neural network there are three types of units:**

- Input units, which receive signals from the outside.
- Output units send data externally.
- Its hidden units, inputs, and outputs are in the network.

### h. Activation function

- **Activation functions are used in the topic:** [5]

- Relu:
- Softmax:
- Sigmoid:

**Using api tensorflow keras to create neural networks:**

```
model = tf.keras.Sequential()
model.add(layers.Dense(100, input_dim=4, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(16, activation='softmax'))
model.add(layers.Dense(2, activation='sigmoid'))
model.summary()
```

1: Initial model Sequential() - It's called "*sequential*" as it involves identifying a Sequential layer and adding each layer to the model in a linear way, from input to output. [6]

2: Dense represents a fully connected *layer*, which means that all units of the previous layer are connected to the entire unit of the current layer.

The first layer with 100 nodes has an input value of 4 corresponding to 4 input values, with the relu delineated activation function. Next a layer with 64 neurons and then 16 neurons with softmax jaws. The problem produces output with 2 layers, using the sigmoid function for smoother results in line with the binary classification problem.

**i. Loss function in classification**

In essence, a loss function is a function that allows you to determine how different the predicted result is from the actual price to be predicted [7]. It is a way of evaluating the prediction model's quality on the observation dataset. The loss function has a higher value if the prediction model is incorrect, and a smaller value if the prediction model is correct.

**Binary Cross Entropy Loss**

The function only categorizes 2 results 0 and 1 (yes or no). Applied in this topic are 0 off, 1 turn on fan and light bulb.

Condition	
Turn on	Turn off
1	0

**j. Train model**

- **The model will conduct machine learning from input data. Depending on the training algorithm, different models will be produced.**

```
model.fit(X_train,y_train,epochs=50,
batch_size=20,validation_data=(X_val, y_val))
```

- **Train model with 50 train appearances**

It is not possible to put all the data into training in 1 epoch, so we need to divide the dataset into parts (number of batches), each of which is batch size.

- **Compare the aggregate of the predicted results with the different functions.**

**Survey network configuration:**

- *Neural network structure:*
- + Input with 100 neurons. The middle layer has two cases of using the activation function to survey: in order, the RELU function with the number of neurons is 64, followed by the SOFRMAX function with a neuron number of 16.
- + Output with a number of neurons is 2
- *Batch sizes and epochs: 20 and 50*
- *Survey data: Datatraining (50%), data validation (20%), Data test (30%)*

**Table 1 Function comparison table**

Function (Activations   loss   Optimizer)	Data training		Data validation		Data test	
	Acc	Loss	Acc	Loss	Acc	Loss
Relu, relu, softmax   binary_crossentropy   Adam	0.94	0.05	0.98	0.069	0.9	0.06
Relu, relu, softmax   categorical_crossentropy   Adam	0.78	0.29	0.78	0.27	0.7	0.27
Relu, relu   binary_crossentropy   Adam	0.76	0.01	0.71	0.023	0.7	0.025
Relu, relu, softmax   binary_crossentropy   SGD	0.89	0.14	0.94	0.25	0.9	0.26

Grounded on the table over, it can be seen that corresponding to each function actuated by each network subcaste and the optimal functions, different losses produce different results. With the

below results, with the input configuration 100-64-16-2 corresponding to the functions that spark RELU-RELU-SOFTMAX, the loss function binary\_crossentropy and adam optimal function for the stylish results.

### Setup gateway

#### a. Convert model

[8] After having a model by model training, proceed to transfer the TF model to TF lite by command:

```
import tensorflow as tf
modelconverter = tf.lite.TFLiteConverter.from_saved_model(
    saved_model_dir)
flite_model = converter.convert
with open('model.tflite', 'wb') as f: f.write(tflite_model)
```

#### Convert mode tf lite to array C

[9] The following unix command will create a C source file containing TensorFlow Lite models that are a char array:

```
xxd -i model.tflite > model_data.h
```

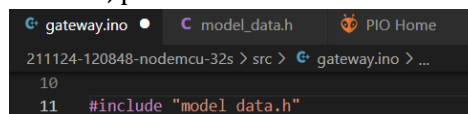
**Result:** file model\_data.h

```
unsigned char model_tflite[] = {
    0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00, 0x04,
    0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00, 0x00, 0x00, 0x18, 0x00,
    0x1c, 0x00, 0x14, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00,
    0x00, 0x24, 0x00, 0x00, 0x00, 0xc4, 0x00, 0x00, 0x00,
    .....
};
unsigned int model_tflite_len = 32712;
```

#### b. Setup model predict on gateway

Include file model convert

After transferring the tflite model file, proceed to install and the child enters esp32's program.



```
gateway.ino • C model_data.h PIO Home
211124-120848-nodemcu-32s > src > gateway.ino > ...
10
11 #include "model_data.h"
```

Figure 8 Include file model in program gateway

Directory structure:

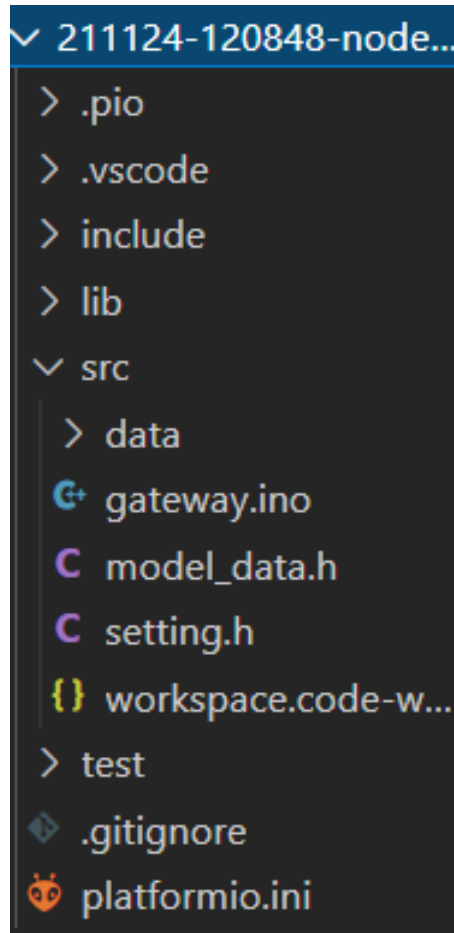


Figure 9 Program gateway directory structure

### c. Model predict

- **Library**

The *EloquentTinyML.h* [10] library is a library developed to make it simpler to install machine learning models on microconditions.

This library is compatible with **all architectures**, it is possible to use it on all Arduino boards. Libraries can be installed from download files on the website, github or in the library manager of arduino IDE.

- **Install a predictive program** [11]

#### Add a library

```
#include "EloquentTinyML.h"  
#include "model_data.h"
```

#### Install

```
#define TENSOR_ARENA_SIZE 34 * 1024  
Eloquent::TinyML::TfLite<NUMBER_OF_INPUTS,    NUMBER_OF_OUTPUTS,  
TENSOR_ARENA_SIZE> ml;
```

Declare the size of the model file and initialize the predicted variable for the Eloquent library

```
ml.begin(model_tflite);
```



Call the model array storage variable in the converted model file to C array.

*Predict*

```
float input[4] = {tem, hum, qtro, people};  
//Serial.println(input[3]);  
float output[2] = {0, 0};  
ml.predict(input, output);
```

Declare input data for the prediction model with values in order as in the training model section.

Declare the output array with 2 values, *output[0]* corresponding to the lamp, *output[1]* corresponding to the value of the fan. With the binary prediction model producing results in the range [0.1], in the case of turning the device on and off, the device will select 0.5 to set the on and off condition. With a predicted result of less than 0.5, the opposite will be turned off and the opposite is greater than 0.5 will turn on.

```
void mode_auto(){  
  float input[4] = {tem, hum, qtro, people};  
  float output[2] = {0, 0};  
  ml.predict(input, output);  
  String lamp = "";  
  String fan = "";  
  if (output[0] > 0.5){  
    digitalWrite(13, LOW);  
    datarx[0] = 1;  
    lamp = "true";  
  }  
  if (output[1] > 0.5){  
    digitalWrite(14, LOW);  
    datarx[1] = 1;  
    fan = "true";  
  }  
  if (output[0] < 0.5){  
    digitalWrite(13, HIGH);  
    datarx[0] = 0;  
    lamp = "false";  
  }  
  if (output[1] < 0.5)  
  {  
    digitalWrite(14, HIGH);  
    datarx[1] = 0;  
    fan = "false";  
  }  
  client.publish("danhtran1998/esp32/lamp", lamp.c_str());  
  client.publish("danhtran1998/esp32/fan", fan.c_str());  
}
```

**Operating model:**

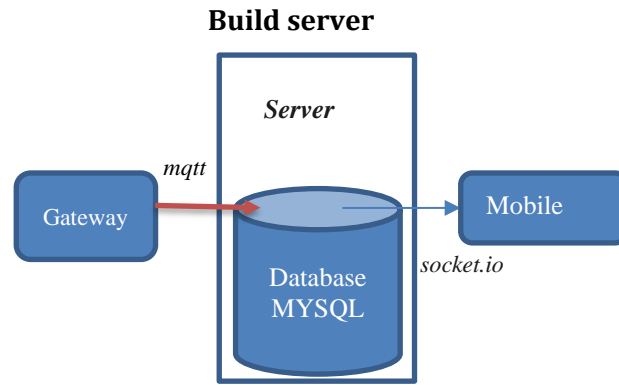


Figure 10 Server operating model

The server receives data from the gateway with the MQTT protocol, all data from the sensor stored by the mysql database. The status of the lights and fans is pushed directly to the mobile app. The sensor data is queried from the database and then pushed to the mobile app. All data is transmitted by socket.io.

**Databasse**

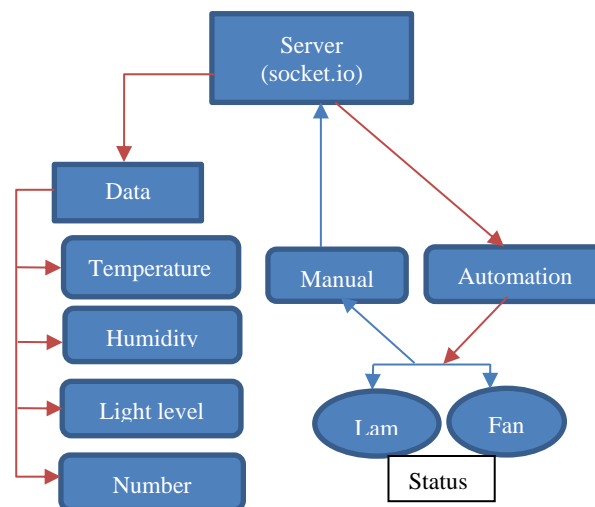
Data table structure of sensors:

#	Name	Type
1	ID	int(50)
2	temp	varchar(50)
3	hum	varchar(50)
4	light	int(50)
5	count	int(50)

Figure 11 Data table structure

Use mysql databases to store sensor data: temperature, humidity, light, and number of people entering and leaving the room. Then the server queries and displays to the user app.

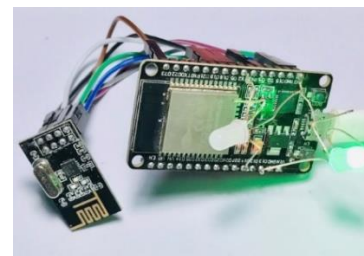
**Build app mobile  
System model**



*Figure 12 System model mobile app*

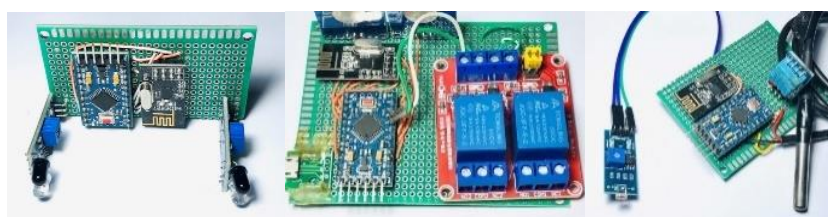
**In the mobile application interface consists of components:**

- Monitor temperature, humidity, room lighting and number of people in the room
- The key component is manual and automation functionality. After selecting the automation function, the mode is sent from the server to the gateway. The automation part is operated by the machine learning model available on the esp32 micro-control module. Then the status of 2 fan and light devices is sent back to the server, from which the server sends back the mobile app corresponding to 2 switches.
- The rest is the device on and off switch, which will be active when the mode switch switches to manual status.
- On the app is transmitted and received by socket.io.



*Figure 13 Realistic interface image*

*Figure 14 Hardware gateway image*



*Figure 15 Hardware count people image - node switch fan & lamp - node sensor temperature, humidity & light*

**CONCLUDE**

Applying machine learning to an edge device can help improve the service provided in IoT applications. ONE prototype of IoT-based home automation system presented in this paper, applying deep learning edge device intelligence. This paper has presented a system that applies machine learning and edge computing models, the system operates stably with a fairly high rate and accuracy up to 0.9848 with the test data set. And the way to implement machine learning model on microcontroller is ESP32, in order to reduce the processing time and speed of server actions. In addition, the system is monitored using a mobile app, and lights and fans can be

controlled when manual mode is selected in the app. Besides, it has solved the internet bandwidth problem and if the connection is lost it can be fully automatic with RF24 radio network. However, the model is still passive because the data cannot learn and update regularly to take appropriate actions.

## REFERENCES

- [1] <https://builtin.com/artificial-intelligence/artificial-intelligence-automotive-industry>.
- [2] S. K. D. U. K. A. Khandaker Mohammad Mohi Uddin, "MirrorME: implementation of an IoT based smart mirror through facial recognition and personalized information recommendation algorithm," 12 9 2021.
- [3] Y. W. S. C. X. L. Xuehao Shen, "An Intelligent Garbage Sorting System Based on Edge Computing and Visual Understanding of Social Internet of Vehicles," 31 8 2021.
- [4] Saleem, Tausifa & Chishti, Ahsan, "Data Analytics in the Internet of Things: A Survey," *Scalable Computing: Practice and Experience*, no. 20, pp. 607-630, 2019.
- [5] Sharma, S., Sharma, S. and Athaiya, A., 2017. Activation functions in neural networks. *towards data science*, 6(12), pp.310-316.
- [6] [https://www.tensorflow.org/api\\_docs/python/tf/keras/Sequential](https://www.tensorflow.org/api_docs/python/tf/keras/Sequential).
- [7] Ho, Y. and Wookey, S., 2019. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8, pp.4806-4813.
- [8] TensorFlow. 2021. *TensorFlow Lite converter*. [online] Available at: <<https://www.tensorflow.org/lite/convert>>
- [9] tensorflow. 2021. *Build and convert models*. [online] Available at: <[https://www.tensorflow.org/lite/microcontrollers/build\\_convert](https://www.tensorflow.org/lite/microcontrollers/build_convert)>
- [10] Yusuf, A.S., Iffah, P.Y.D., Pauzi, G.A., Surtono, A., Suciayati, S.W. and Triyana, K., 2019, October. Flow Rate and Volume Control of Fluid Based on Arduino for Synthesis of Silver Nanowires. In *Journal of Physics: Conference Series* (Vol. 1338, No. 1, p. 012018). IOP Publishing.
- [11] Ray, P.P., 2021. A Review on TinyML: State-of-the-art and Prospects. *Journal of King Saud University-Computer and Information Sciences*.
- [12] T. J. Saleem, "Deep learning for the internet of things: Potential benefits and use-cases," 8 12 2020.